



DIGITAL RESEARCH®

Post Office Box 579
160 Central Avenue
Pacific Grove, California 93950
(408) 649-3896
TWX 910 360 5001

CBASIC To CB-80 Conversion Aid January 1982

This conversion aid is provided to help convert your CBASIC programs to CB-80 version 1.2. CB-80 and LK-80 will operate on any CP/M or MP/M system. However, output (i.e. Composite Programs) from CB-80 and LK-80 require CP/M version 2 or MP/M. When compiling your source code in CB-80, it is important to pay close attention to all error messages produced. This is the fastest way to determine any necessary changes. Most programs will recompile with no conversion. If any problems arise, please contact the Digital Research hotline (408-375-6262) for assistance.

SUBSCRIPTED VARIABLES (Arrays)

CBASIC allows a dimensioned variable name (an array) to also be used as a simple or unsubscripted variable. CBASIC treated these as separate and distinct variables. CB-80 does not allow a dimensioned variable to be referenced without the array index.

Example:

CBASIC

```
DIM A% (20)

FOR I% = 1 TO 20
  A% (I%) = 0
NEXT I%

A% = 100
```

CB-80

```
DIM A% (20)

FOR I% = 1 TO 20
  A% (I%) = 0
NEXT I%

A% = 100
```

(Error message #36)

CB-80 issues error message #36 (an identifier is used as a simple variable with previous usage as a subscripted variable) for the statement A% = 100.

CONFIDENTIAL

Page 1

10-10-60

CONFIDENTIAL

January 1961

This document contains information which is being furnished to you for your information only. It is not to be distributed outside your organization. The information contained herein is the property of the United States Government and is loaned to you. It and its contents are not to be distributed outside your organization. The information contained herein is the property of the United States Government and is loaned to you. It and its contents are not to be distributed outside your organization.

The information contained herein is the property of the United States Government and is loaned to you. It and its contents are not to be distributed outside your organization. The information contained herein is the property of the United States Government and is loaned to you. It and its contents are not to be distributed outside your organization.

(S)

This document contains information which is being furnished to you for your information only. It is not to be distributed outside your organization. The information contained herein is the property of the United States Government and is loaned to you. It and its contents are not to be distributed outside your organization.

Example: <u>CBASIC</u> A% = 100 DIM A% (20) FOR I% = 1 TO 20 A% (I%) = 0 NEXT I% END	<u>CB-80</u> A% = 100 DIM A% (20) (Error message #93) FOR I% = 1 TO 20 A% (I%) = 0 (Error message #37) NEXT I%
---	--

CB-80 issues error message #93 (a variable in a DIM statement is defined previously as other than a subscripted variable) for the statement DIM A% (20). Also, CB-80 issues error message #37 (an identifier is used as a subscripted variable with previous usage as an unsubscripted variable) for the statement A% = 100.

This is corrected by changing the unsubscripted variable to a different variable name of the same type. Be careful that the new variable name chosen is distinct from all other variable names used in your program.

FILE STATEMENT

The FILE statement in CBASIC opens a file present on the referenced disk; otherwise a file with the specified name is created. CB-80 does not implement the FILE statement, but the same action as the FILE statement is accomplished by using the OPEN, SIZE and CREATE statements.

Example: <u>CBASIC</u> FILE NAME\$	<u>CB-80</u> IF SIZE (NAME\$) <> 0 \ THEN OPEN NAME\$ AS FILE.NO% \ ELSE CREATE NAME\$ AS FILE.NO%
--	--

In the CB-80 example, if the file NAME\$ exists, then the file is opened as usual. If the file does not exist, or its length is zero (as determined by the SIZE statement), then the IF statement passes control to the CREATE statement which creates the file NAME\$. Please note that the OPEN and CREATE statements require a file reference number (FILE.NO%); the FILE statement does not require one.

...of the ...
...the ...
...the ...

...the ...
...the ...
...the ...

...the ...
...the ...
...the ...

...the ...
...the ...
...the ...

...the ...
...the ...
...the ...

When converting a FILE statement, care should be taken to ensure the file number chosen does not conflict with any other file reference numbers already used in your program, and that PRINT and READ statements accessing the file are modified to reflect the new file number.

SAVEMEM

The SAVEMEM statement, used in CBASIC to execute routines written in assembler, has no meaning in CB-80. Use of assembler routines and how they can be linked into CB-80 programs is discussed in the LK-80 Operator's Guide and in the CB-80 Language Manual.

CHAIN STATEMENT

The CHAIN statement in CBASIC and CB-80 passes control from the program currently executing in memory to the program selected in the CHAIN statement. The format of the CHAIN statement is the same in CBASIC and CB-80.

Example:	<u>CBASIC</u>	<u>CB-80</u>
	CHAIN <expression>	CHAIN <expression>

The expression must evaluate to an unambiguous file name on the disk. If the file name selected in the expression does not include the file name extension, CBASIC assumes a .INT type file; CB-80 assumes an .OVL (overlay) type file.

The OVL type file in CB-80 is not the root of a chaining sequence. The root program has a .COM extension as part of the file name. If your program is chaining back to the original root (.COM file) or a different root. The expression in the CHAIN statement must evaluate to a file name with a .COM extension. A CB-80 program can chain to a .COM file other than the one generated by CB-80 and LK-80.

STRING LENGTHS

String lengths up to 32k bytes are allowed in CB-80. To provide this expanded string length, CB-80 uses two bytes for the string length whereas CBASIC uses one byte.

If your program uses the SADD function in conjunction with PEEK and POKE to pass a string to an assembly language routine, you will have to make necessary changes in your program to accommodate the two byte length indicator in CB-80.

Example:

CBASIC

CB-80

```
LEN% = PEEK (SADD(STRING$))  LEN% = (PEEK (SADD(STRING$)) AND 07FH \
END                          + PEEK (SADD(STRING$) + 1)) * 256
```

PEEK AND POKE

The PEEK function in CBASIC and CB-80 returns the contents of the memory location specified in the PEEK function call. Memory locations in CB-80 do not necessarily contain the same information that CBASIC programs expected to find. You may have to change the memory location your program is examining or remove the PEEK statement from your program.

The POKE statement behaves in the same manner in CB-80 as it does in CBASIC. However, the memory locations in CB-80 are not the same as the memory locations in CBASIC. If your program contains a POKE statement to a location in a CBASIC program, it may have a disastrous effect when used in a CB-80 program. In particular, the statement:

```
POKE 0110H, 0
      or
POKE 272, 0
```

used in CBASIC to adjust the console width must be removed.

Because the actual location of code is determined by the linker, extreme care must be taken when using the POKE statement.

FOR-NEXT LOOPS

When using nested FOR-NEXT loops in CBASIC, the NEXT statement is allowed to terminate more than one loop. CB-80 does not allow this construct. A separate NEXT statement must be used for each FOR statement which begins a loop.

Example: <u>CBASIC</u> <pre> FOR I% = 1 TO 100 FOR J% = 1 TO 100 . . (statements) . NEXT J%, I%</pre>	<u>CB-80</u> <pre> FOR I% = 1 TO 100 FOR J% = 1 TO 100 . . (statements) . NEXT J% NEXT I%</pre>
--	--

Also, CBASIC executes all statements in the FOR-NEXT loop at least once. CB-80 executes the statements in a FOR-NEXT loop zero or more times depending on the values of the loop indexes. This could be a potential problem. Examine the logic of your programs and make any necessary changes.

CONSOLE WIDTH

In order to facilitate cursor addressing, CB-80 generates a carriage return only upon executing a PRINT statement not terminated by a comma or semicolon (analogous to setting the CBASIC console width to zero by a POKE to 272 (110h)), rather than automatically generating a carriage return when the console width has been exceeded, as in CBASIC. As a result, CBASIC programs which assume that the cursor returns when the console width is exceeded may not execute correctly.

FRE

In CB-80, FRE returns a binary value representing the number of bytes of available memory, rather than a real value as in CBASIC. Since CB-80 arithmetic routines interpret binary values in excess of 32,767 as negative numbers, programs using FRE must interpret "negative" values correctly. In general, negative values indicate ample available memory.

The following statement may be used to determine whether adequate memory is available:

```

IF (FRE > 0) AND (FRE < MIN.MEMORY%) THEN \
    CALL LOW.MEMORY.WARNING
```

READ AND INPUT STATEMENTS FOR INTEGERS

READ and INPUT statements handle integers differently in the two languages. CBASIC accepts all numeric values as real

MEMORANDUM FOR THE DIRECTOR, FBI

DATE: 10/10/68

TO: DIRECTOR, FBI

FROM: SAC, NEW YORK

SUBJECT: [Illegible]

On 10/10/68, [illegible] advised that [illegible] had been [illegible] by [illegible] on 10/10/68. [illegible] advised that [illegible] had been [illegible] by [illegible] on 10/10/68. [illegible] advised that [illegible] had been [illegible] by [illegible] on 10/10/68.

Very truly yours,

[Illegible signature and text block]

On 10/10/68, [illegible] advised that [illegible] had been [illegible] by [illegible] on 10/10/68. [illegible] advised that [illegible] had been [illegible] by [illegible] on 10/10/68. [illegible] advised that [illegible] had been [illegible] by [illegible] on 10/10/68.

The following information was obtained from [illegible] on 10/10/68.

[Illegible signature and text block]

READ AND CONFIDENTIAL - SECURITY INFORMATION

numbers, and then converts to integers if required. CB-80 accepts integers directly.

Example:	<u>CBASIC</u>	<u>CB-80</u>
	DATA 10.7, 1E2	DATA 10.7, 1E2
	READ A%,B%	READ A%,B%

The values of A% and B%
after the READ are:

A% = 11 B% = 100

The values of A% and B%
after the READ are:

A% = 10 B% = 1

With CB-80, conversion stops at the first character not a part of a valid integer.

FUNCTION AND VARIABLE NAMES

CB-80 requires function names, variables and statement labels to be unique. This should not create problems in converting CBASIC programs to CB-80 since CBASIC required all functions to start with the letters "FN", and labels can only be numeric constants. Remember that variables and arrays may conflict as described above.

LABELS

All labels in a program are placed in the symbol table, including the ones not referenced. CBASIC did not put unreferenced labels in the symbol table.

A label in a multiple line function is local to the function. This is not the same in CBASIC.

Example:	<u>CBASIC</u>	<u>CB-80</u>
	DEF FN.A	DEF FN.A
	100 PRINT "HELLO"	100 PRINT "HELLO"
	FEND	FEND
	GOTO 100	GOTO 100

(Error message #82)

CB-80 issues error message #82 (the label in a GOTO statement is not defined; the label used in a function must be defined in that function).

numbers, and their conversion to integers is described in the CR-80 manual.

Example:
CR-80: `DATA 100, 100`
CR-85: `DATA 100, 100`
CR-80: `READ A, B`
CR-85: `READ A, B`

The values of A and B are 100 and 100. After the READ statement, the values of A and B are 100 and 100.

With CR-80, arrays for arrays of the type described in the CR-80 manual.

FUNCTIONS AND VARIABLES

CR-80 defines several functions and variables. These are defined in the CR-80 manual. The CR-85 manual defines the same functions and variables, but with some changes. The CR-85 manual defines the same functions and variables, but with some changes.

Labels:
All labels in a program are placed in the label table. The label table is a table of labels and their addresses. The label table is a table of labels and their addresses.

Example:
CR-80: `GO TO 100`
CR-85: `GO TO 100`
CR-80: `GO TO 100`
CR-85: `GO TO 100`

CR-80 uses a label to refer to a label in the CR-80 manual. The label is not defined in the CR-80 manual. The label is not defined in the CR-80 manual.

WARNING MESSAGES

There are no warning messages produced during the execution of a CB-80 program. All errors are fatal and execution will terminate unless an ON ERROR GOTO statement is used to trap the error.

NEW RESERVED WORDS

CB-80 has incorporated several new reserved words with some of the newly implemented features. If your CBASIC programs use these words as variables, rename them to a different variable name. The reserved words unique to CB-80 are listed below:

ERR
EXTERNAL
INTEGER
MOD
READONLY
UNLOCK

ERRL
GET
LOCK
PUBLIC
REAL
UNLOCKED

ERROR
INKEY
LOCKED
PUT
STRING

ENHANCEMENTS IN CB-80

Much time and effort went into the development of CB-80. Besides making CB-80 a true compiler version of CBASIC, many internal changes are implemented. CB-80 incorporates some new features which are not included in CBASIC. These new features are listed below. The page numbers in the CB-80 Language Manual listed alongside the features will give you more detail on how you can implement these features into your programs.

<u>Enhancements</u>	<u>Page # In CB-80 Manual</u>
32K byte strings	19
Alpha-numeric labels	12, 13
Assembly language routines	105
ATTACH function	*
Binary file IO with GET, PUT	89, 91
CHAIN statement will load any COM file	106, 107
DETACH statement	*
EXTERNAL and PUBLIC functions	104
INKEY function	78
Local variables in functions	30
LOCK and UNLOCK functions	91, 93
MFRE function	51, 82
MOD function	45
Nested IF statements	54
ON ERROR statement, ERR and ERRL functions	51, 65
Variable type declarations	21, 22, 23

* Will appear in the next printing of the CB-80 manual



DIGITAL RESEARCH LANGUAGES END USER LICENSE AGREEMENT

*Use and possession of this software package
is governed by the following terms.*

1. DEFINITIONS - These definitions shall govern:

- A. "DRI" means DIGITAL RESEARCH INC., P.O. Box 579, Pacific Grove, California 93950, the author and owner of the copyright on this SOFTWARE.
- B. "CUSTOMER" means the individual purchaser and the company CUSTOMER works for, if the company paid for this SOFTWARE.
- C. "COMPUTER" is the single micro-computer on which CUSTOMER uses this program. Multiple CPU systems may require supplementary licenses.
- D. "SOFTWARE" is the set of computer programs in this package, regardless of the form in which CUSTOMER may subsequently use it, and regardless of any modification which CUSTOMER may make to it.
- E. "LICENSE" means this Agreement and the rights and obligations which it creates under the United States Copyright Law and California laws.
- F. "RUNTIME LIBRARY" is the set of copyrighted DRI language sub-routines, provided with each language compiler, a portion of which must be linked to and become part of a Customer program for that program to run on the COMPUTER.

2. LICENSE

DRI grants CUSTOMER the right to use this serialized copy of the SOFTWARE on a single COMPUTER at a single location so long as CUSTOMER complies with the terms of the LICENSE, and either destroys or returns the SOFTWARE when CUSTOMER no longer has this right. CUSTOMER may not transfer the program electronically from

one computer to another over a network. DRI shall have the right to terminate this license if CUSTOMER violates any of its provisions. CUSTOMER owns the diskette(s) purchased, but under the Copyright Law DRI continues to own the SOFTWARE recorded on it and all copies of it. CUSTOMER agrees to make no more than five (5) copies of the SOFTWARE for backup purposes and to place a label on the outside of each backup diskette showing the serial number, program name, version number and the DRI copyright and trademark notices in the same form as the original copy. CUSTOMER agrees to pay for licenses for additional user copies of the SOFTWARE if CUSTOMER intends to or does use it on more than one COMPUTER. If the micro-computer on which CUSTOMER uses the SOFTWARE is a multi-user microcomputer system, then the license covers all users on that single system, without further license payments, only if the SOFTWARE is used only on that microcomputer. This is NOT a license to use the SOFTWARE on mainframes or emulators.

3. TRANSFER OR REPRODUCTION

CUSTOMER understands that unauthorized reproduction of copies of the SOFTWARE and/or unauthorized transfer of any copy may be a serious crime, as well as subjecting CUSTOMER to damages and attorney fees. CUSTOMER may not transfer any copy of the SOFTWARE to another person unless CUSTOMER transfers all copies, including the original, and advises DRI of the name and address of that person, who must sign a copy of the registration card, pay the then current transfer fee, and agree to the terms of this LICENSE in order to use the SOFTWARE. DRI will provide additional copies of the card and LICENSE upon request. DRI has the right to terminate the LICENSE, to trace serial numbers, and to take legal action if these conditions are violated.

4. COMPOSITE PROGRAMS

As an exception to Paragraph 3, CUSTOMER is granted the right to include portions of the DRI RUNTIME LIBRARY in CUSTOMER developed programs, called COMPOSITE PROGRAMS, and to use, distribute and license such COMPOSITE PROGRAMS to third parties without payment of any further license fee. CUSTOMER shall, however, include in such COMPOSITE PROGRAM, and on the exterior label of every diskette, a copyright notice in this form: "Portions of this program, ©1982 DIGITAL RESEARCH INC." In cases where such COMPOSITE PROGRAM is contained in READ-ONLY-MEMORY (ROM) chips, a copyright notice in the form listed above, must be displayed on the exterior of the chip and internally in the chip (in ASCII literal form). As an express condition to the use of the RUNTIME LIBRARY, CUSTOMER agrees to indemnify and hold DRI harmless from all claims by CUSTOMER and third parties arising out of the use of COMPOSITE PROGRAMS.

5. LIMITED WARRANTY

The only warranty DRI makes is that the diskette(s) on which the SOFTWARE is recorded will be replaced without charge, if DRI in good faith determines that the media was defective and not subject to misuse, and if returned to DRI or the dealer from whom it was purchased, with a copy of the original registration card, within ten days of purchase. Customer will receive support from the Vendor from whom customer has purchased the software. In addition, support is available from DRI directly, for qualified, registered customers under DRI's then current support policies. DRI reserves the right to change the specifications and operating characteristics of the SOFTWARE it produces, over a period of time, without notice.

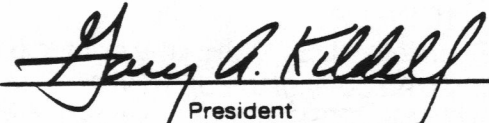
6. DRI MAKES NO OTHER WARRANTIES, EITHER EXPRESSED OR IMPLIED, AND DRI SHALL NOT BE LIABLE FOR WARRANTIES OF FITNESS OF PURPOSE

OR MERCHANTABILITY, NOR FOR INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES SUCH AS LOSS OF PROFITS OR INABILITY TO USE THE SOFTWARE. SOME STATES MAY NOT ALLOW THIS DISCLAIMER SO THIS LANGUAGE MAY NOT APPLY TO CUSTOMER. IN SUCH CASE, OUR LIABILITY SHALL BE LIMITED TO REFUND OF THE DRI LIST PRICE. CUSTOMER MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE. CUSTOMER and DRI agree that this product is not intended as "Consumer Goods" under state or federal warranty laws.

7. MISCELLANEOUS

This is the only agreement between CUSTOMER and DRI and it cannot and shall not be modified by purchase orders, advertising or other representations of anyone, unless a written amendment has been signed by one of our company officers. When CUSTOMER opens the SOFTWARE package or uses the SOFTWARE, this act shall be considered as mutual agreement to the terms of this LICENSE. This LICENSE shall be governed by California law, except as to copyright matters which are covered by Federal laws, and is deemed entered into at Pacific Grove, Monterey County, CA by both parties.


**DIGITAL
RESEARCH™**

by 
President

SAVE THIS LICENSE FOR FUTURE REFERENCE

Appendix C

ASCII CHARACTER CODES AND BASIC-E CHARACTER SET

ASCII CHARACTER CODES

Character	Decimal	Ctrl.	Hex	Binary	B	Mnemonic Expansion
NUL	0		0	0000000		Null
SOH	1	A	1	0000001		Start Of Heading
STX	2	B	2	0000010		Start of Text
ETX	3	C	3	0000011		End of Text
EOT	4	D	4	0000100		End Of Transmission
ENQ	5	E	5	0000101		ENquiry
ACK	6	F	6	0000110		ACKnowledge
BEL	7	G	7	0000111		BELl
BS	8	H	9	0001000		Back Space
HT	9	I	9	0001001		Horizontal Tab
LF	10	J	A	0001010		Line Feed
VT	11	K	B	0001011		Vertical Tab
FF	12	L	C	0001100		Form Feed
CR	13	M	D	0001101	+	Carriage Return
SO	14	N	E	0001110		Shift Out
SI	15	O	F	0001111		Shift In
DLE	16	P	10	0010000		Data Link Escape
DC1	17	Q	11	0010000		Peripheral Control
DC2	18	R	12	0010010		Peripheral Control
DC3	19	S	13	0010011		Peripheral Control
DC4	20	T	14	0010100		Peripheral Control
NAK	21	U	15	0010101		Not Acknowledged
SYN	22	V	16	0010110		SYNchronous Idle
ETB	23	W	17	0010111		End Transmission Block
CAN	24	X	18	0011000		CANcel
EM	25	Y	19	0011001		End of Medium
SUB	26	Z	1A	0011010		SUBstitute
ESC	27		1B	0011011		ESCape
FS	28		1C	0011100		File Separator
GS	29		1D	0011101		Group Separator
RS	30		1E	0011110		Record Separator
US	31		1F	0011111		Unit Separator
Space	32		20	0100000	+	Space (blank)
!	33		21	0100001		Exclamation Point
"	34		22	0100010	+	Quotation Mark
#	35		23	0100011	+	Number Sign
\$	36		24	0100100	+	Dollar Sign
%	37		25	0100101		Per-cent Sign
&	38		26	0100110		Ampersand
'	39		27	0100111		Apostrophe
(	40		28	0101000	+	Open Parenthese
)	41		29	0101001	+	Close Parenthese
*	42		2A	0101010	+	Asterisk
+	43		2B	0101011	+	Plus
,	44		2C	0101100	+	Comma

<u>Character</u>	<u>Decimal</u>	<u>Ctrl.</u>	<u>Hex</u>	<u>Binary</u>	<u>B</u>	<u>Mnemonic Expansion</u>
-	45		2D	0101101	+	Minus
.	46		2E	0101110	+	Period
/	47		2F	0101111	+	Slash
0	48		30	0110000	+	Digit
1	49		31	0110001	+	Digit
2	50		32	0110010	+	Digit
3	51		33	0110011	+	Digit
4	52		34	0110100	+	Digit
5	53		35	0110101	+	Digit
6	54		36	0110110	+	Digit
7	55		37	0110111	+	Digit
8	56		38	0111000	+	Digit
9	57		39	0111001	+	Digit
:	58		3A	0111010	+	Colon
;	59		3B	0111011	+	Semi-colon
<	60		3C	0111100	+	Less than
=	61		3D	0111101	+	Equal
>	62		3E	0111110	+	Greater than
?	63		3F	0111111		Question Mark
@	64		40	1000000		At Sign
¹⁰ A	65		41	1000001	+	Upper Case
¹¹ B	66		42	1000010	+	Upper Case
¹² C	67		43	1000011	+	Upper Case
¹³ D	68		44	1000100	+	Upper Case
¹⁴ E	69		45	1000101	+	Upper Case
¹⁵ F	70		46	1000110	+	Upper Case
¹⁶ G	71		47	1000111	+	Upper Case
¹⁷ H	72		48	1001000	+	Upper Case
¹⁸ I	73		49	1001001	+	Upper Case
¹⁹ J	74		4A	1001010	+	Upper Case
²⁰ K	75		4B	1001011	+	Upper Case
²¹ L	76		4C	1001100	+	Upper Case
²² M	77		4D	1001101	+	Upper Case
²³ N	78		4E	1001110	+	Upper Case
²⁴ O	79		4F	1001111	+	Upper Case
²⁵ P	80		50	1010000	+	Upper Case
²⁶ Q	81		51	1010001	+	Upper Case
²⁷ R	82		52	1010010	+	Upper Case
²⁸ S	83		53	1010011	+	Upper Case
²⁹ T	84		54	1010100	+	Upper Case
³⁰ U	85		55	1010101	+	Upper Case
³¹ V	86		56	1010110	+	Upper Case
³² W	87		57	1010111	+	Upper Case
³³ X	88		58	1011000	+	Upper Case
³⁴ Y	89		59	1011001	+	Upper Case
³⁵ Z	90		5A	1011010	+	Upper Case
[	91		5B	1011011		Open Bracket
\	92		5C	1011100	+	Back-Slash
]	93		5D	1011101		Closed Bracket

37 3C 38 6E

Character	Decimal	Ctrl.	Hex	Binary	B	Mnemonic Expansion
+ or ^	94		5E	1011110	+	Up arrow (circumflex)
+ or _	95		5F	1011111		Left arrow (underscore)
- or -	96		60	1100000		Not Sign
a	97		61	1100001		Lower Case
b	98		62	1100010		Lower Case
c	99		63	1100011		Lower Case
d	100		64	1100100		Lower Case
e	101		65	1100101		Lower Case
f	102		66	1100110		Lower Case
g	103		67	1100111		Lower Case
h	104		68	1101000		Lower Case
i	105		69	1101001		Lower Case
j	106		6A	1101010		Lower Case
k	107		6B	1101011		Lower Case
l	108		6C	1101100		Lower Case
m	109		6D	1101101		Lower Case
n	110		6E	1101110		Lower Case
o	111		6F	1101111		Lower Case
p	112		70	1110000		Lower Case
q	113		71	1110001		Lower Case
r	114		72	1110010		Lower Case
s	115		73	1110011		Lower Case
t	116		74	1110100		Lower Case
u	117		75	1110101		Lower Case
v	118		76	1110110		Lower Case
w	119		77	1110111		Lower Case
x	120		78	1111000		Lower Case
y	121		79	1111001		Lower Case
z	122		7A	1111010		Lower Case
{	123		7B	1111011		Open Brace
	124		7C	1111100		Bar
}	125		7D	1111101		Closed Brace
~	126		7E	1111110		Tilde
DEL	127		7F	1111111		Delete

* A + in the B column means that the character is part of the BASIC-E character set. (Lower case is acceptable but not standard.)

The American Standard Coded Information Interchange is an arbitrary set of bit configurations in a byte, which computer users have agreed to use to represent alpha-numeric data. The bits used are the 0 - 6 bits. The (most significant) 7 bit is not used in 8080 processing. (Bits are numbered from right to left starting with zero.)

The 7 bits can also be looked upon as if they were integer numbers (0 through 127) and are often referenced by this apparent numeric representation of the arbitrary code (see function ASC).

The program has to know whether this byte that it receives is numeric or ASCII (string) data, because a byte is merely a set of 8 bits (each representable by 0 or 1). What it means depends on the hardware and software in the computer. Mode is always a programming consideration.

Some systems use the ASCII character byte's most significant bit as a parity check bit. Normally, BASIC-E does not check for parity. "Odd parity" implies an odd number of 1 bits, and "even parity" implies an even number of 1 bits in each byte. Output routines usually set all parity bits. Input routines can then check the integrity of all ASCII data by checking the bytes for the specified parity.

In the ASCII table, non-printing "characters" are shown as output "codes". Thus, ASCII 7 (produced by a CTRL-G) does not print "BEL"; it is supposed to ring a bell (or sound a buzzer) on output terminals equipped with such devices.

Do not include the line-feed (LF), form-feed (FF), vertical tab (VT), or carriage return (CR) in strings that have any other characters in them. On many systems a form-feed character write halts the printing terminal (to permit paper positioning). The non-printing character set (ASCII 1 through 31) is used in various ways by more or less "intelligent" terminals to control software and hardware operations, including plotting control.

ASCII & IEEE (GPIB) CODE CHART

BITS				0 0		0 1		1 0		1 1		1 1	
B7 B6 B5				0 0		0 1		1 0		1 1		1 1	
B4 B3 B2 B1				CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE			
0 0 0 0				0 NUL	20 DLE	40 SP	60 0	100 @	120 P	140 '	160 p		
0 0 0 1				1 SOH	21 DC1	41 !	61 1	101 A	121 Q	141 a	161 q		
0 0 1 0				2 STX	22 DC2	42 "	62 2	102 B	122 R	142 b	162 r		
0 0 1 1				3 ETX	23 DC3	43 #	63 3	103 C	123 S	143 c	163 s		
0 1 0 0				4 EOT	24 DC4	44 \$	64 4	104 D	124 T	144 d	164 t		
0 1 0 1				5 ENQ	25 NAK	45 %	65 5	105 E	125 U	145 e	165 u		
0 1 1 0				6 ACK	26 SYN	46 &	66 6	106 F	126 V	146 f	166 v		
0 1 1 1				7 BEL	27 ETB	47 ,	67 7	107 G	127 W	147 g	167 w		
1 0 0 0				8 BS	30 CAN	50 (	70 8	110 H	130 X	150 h	170 x		
1 0 0 1				9 HT	31 EM	51)	71 9	111 I	131 Y	151 i	171 y		
1 0 1 0				10 LF	32 SUB	52 *	72 :	112 J	132 Z	152 j	172 z		
1 0 1 1				11 VT	33 ESC	53 +	73 ;	113 K	133 [	153 k	173 {		
1 1 0 0				12 FF	34 FS	54 ,	74 <	114 L	134 \	154 l	174 		
1 1 0 1				13 CR	35 GS	55 -	75 =	115 M	135]	155 m	175 }		
1 1 1 0				14 SO	36 RS	56 .	76 >	116 N	136 ^	156 n	176 ~		
1 1 1 1				15 SI	37 US	57 /	77 ?	117 O	137 _	157 o	177 RUBOUT (DEL)		
				ADDRESSED COMMANDS		UNIVERSAL COMMANDS		LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY octal 25 PPU GPIB code
 hex 15 21 NAK ASCII character
 decimal

Get familiar with ASCII and IEEE-488 codes with the help of this handy table. (Copyright © 1979, Tektronix Inc. All rights reserved. Reproduced by permission.)

